An investigation into determining head pose for gaze estimation on unmodified mobile devices

Stephen Ackland* De Montfort University
Howell Istance† De Montfort University
Simon Coupland‡ De Montfort University
Stephen Vickers§ De Montfort University

Abstract

Traditionally, devices which are able to determine a users gaze are large, expensive and often restrictive. We investigate the prospect of using common webcams and mobile devices such as laptops, tablets and phones without modification as an alternative means for obtaining a users gaze. A person's gaze can be fundamentally determined by the pose of the head as well as the orientation of the eyes. This initial work investigates the first of these factors - an estimate of the 3D head pose (and subsequently the positions of the eye centres) relative to a camera device. Specifically, we seek a low cost algorithm that requires only a one-time calibration for an individual user, that can run in real-time on the aforementioned mobile devices with noisy camera data. We use our head tracker to estimate the 4 eye corners of a user over a 10 second video. We present the results at several different frames per second (fps) to analyse the impact on the tracker with lower quality cameras. We show that our algorithm is efficient enough to run at 75fps on a common laptop, but struggles with tracking loss when the fps is lower than 10fps.

CR Categories: I.4.8 [Image Processing And Computer Vision]: Scene Analysis—[Tracking]

Keywords: Webcam systems, Computer vision, 3D Head Pose

1 Introduction

This work investigates the prospect of incorporating gaze tracking capabilities into mobile devices, particularly laptops, smart phones and tablets. The processing power and memory available on such devices now allows for computationally intensive applications to run in real-time, and coupled with their inclusion of high resolution cameras and inertial measurement sensors gives potential to developing a portable dedicated eye tracking device. Gaze tracking on such devices could provide another modality of input instead of or alongside touch interfaces that have become ubiquitous on such platforms. The accuracy of the eye tracker is influenced heavily by the lens and resolution of the camera on a mobile device. Mobile devices may also have to deal with the extremes of a natural environment, for example varying light conditions and also variability in positioning of the device.

It is known that the two main contributing factors for eye gaze are head pose and eye location relative to the head [Langton et al. 2004]. Most solutions simplify the problem by only considering one and fixing the other, i.e. assuming the head position remains relatively stationary and creating a 2D mapping of the eye position to screen coordinates via a simple calibration procedure, or foregoing the eye location and simply determine the gaze vector via the orientation of the head.

Many of the common 2D mappings use infra-red technologies through the well-documented pupil-glint vector. Since an infra-red source isn't typically available on mobile devices, alternative methodologies have to be found. To combat this alternative mappings have been suggested such as appearance based methods with neural networks, or analysing the vectors between the pupils and eye-corners [Sesma et al. 2012]. Even if infra-red light sources were to be attached to mobile devices in the future, the typical state-of-the-art infra-red methodologies provide an additional limitation which makes it unsuitable for mobile technologies, in that by definition the camera should be able to move about easily, which would require calibration procedures to be performed too frequently to be useful to the average user. To overcome these issues we must look towards gaining knowledge about both fundamental aspects of eye gaze i.e. determining the orientation of the head and subsequently combining knowledge gained from the relative eye locations.

Image processing methods can be used on the data provided by the forward-facing camera to determine the location and orientation of the head and subsequently the eyes. Aggregating knowledge about the user's head pose and eye locations with data about a moving camera from an accelerometer could potentially give an estimate of the user's gaze position. At the very least, a robust head tracker allows for robust tracking of the eye regions including the eye corners. In this work we describe a novel 3D head tracking methodology described as a 2.5D Constrained Local Model, that is suitable for the purpose of quickly and robustly tracking the head in a video stream from a webcam or other relatively low quality camera. In section 2 we look at some of the related literature available in this area. Section 3 describes the methodology of our proposed head tracking algorithm. In section 4 we conduct experiments to show how the tracking algorithm allows us to track the eye corners robustly in a video stream which is often required for gaze estimation algorithms. Finally, section 5 offers some conclusions and ideas for future work.

2 Background

Simple examples of 3D head tracking for purposes of gaze estimation have been seen previously via a Cylindrical Head Model [Valenti et al. 2012]. Since only the relative movement is determined, capturing displacement between the head shape and other items in the environment requires further calibration and additionally, like many tracking applications, the accuracy tends to degenerate over time and requires re-instantiation regularly. Alternatively, models of the head shape and texture can be built beforehand, and subsequently we can attempt to acquire a best-fit solution for the model to the new image. One simple representation of shape is the point distribution model (PDM) [Cootes et al. 2001]. The PDM is capable of creating plausible shapes from a sequence of deformable points that are statistically learnt from a training set of marked up images of the shape. With this knowledge we can estimate relative distances between in-scene objects better, and also reduce cumula-

*e-mail: smackland@dmu.ac.uk
†e-mail: hoi@dmu.ac.uk
‡e-mail: simonc@dmu.ac.uk
§e-mail: svickers@dmu.ac.uk

tive tracking errors by always optimising the fitting to the learned data and using the previous tracking iteration as a guide only.

The analysis of the image data can take two main forms: generative models such as the well known Active Appearance Model [Cootes et al. 2001], which are parameterised textures of the whole face, and discriminative models, which combine a number of local feature detectors (patch texture models) representing each point on the face. The discriminative approach comes down to a simple classification problem. For example, *does this new image patch represent an eye corner, or not?* The Constrained Local Model [Cristinacce and Cootes 2006] is one such approach which involves training small texture patches which correspond to parts of the face such as an eye-corner, or nose-bridge. Each patch searches its local area for a 'best-fit' and the pose update of the face is determined via a least squares approach from all the patch results. The result is further enhanced by ensuring that the face is still constrained by the learned shape-model which describes how a face can move.

If we can determine a solution for the 3D eye/cornea centres from similar shape and texture models, we can obviate the need for infrared light sources or stereo camera solutions. One such possibility for a successful geometric approach with a 3D head tracker would be to make some simplifications in that the eye is spherical and that the pupil or iris is actually a perfect circle residing at some 3D coordinate and orientation. Pirri et al.[2011] were able to obtain an estimate of the line of gaze by first detecting the pupil ellipse within the image, and estimating it's 3D pose. The gaze vector was then defined as the surface normal of this circle. With this goal in mind, we are beginning to investigate ways of obtaining the 3D head pose using relatively simple PDM approaches. Previous work in this area include the 'combined 2D + 3D AAM shape model' [Xiao et al. 2004], where the 2D shape generated from the model is further constrained to satisfy the limit of a 3D PDM. The solution described works with a weak-perspective model on the principle that any 3D shape can be represented in 2D by adding 6 additional parameters and a balancing weight. More recently, a 2.5D AAM was introduced which works with a 3D PDM and 2D texture model under a full perspective camera [Martins et al. 2012].

In the following section, we investigate our own novel 3D head pose tracker and analyse it's effectiveness to determine the location of the eye corners, which is a common requirement for interpolation-based gaze estimation. We seek to show that our approach has the potential for improvement over other similar low-cost video based trackers by being robust to significant head movements with low computational complexity (we show that it can run +75 fps (640x480) on a laptop).

3 Methodology

To track the head efficiently with a stream of video data, we build a novel approach called the 2.5D Constrained Local Model, which is similar to a 2.5D Active Appearance Model [Martins et al. 2012] but replaces the generative texture model with a discriminative model that uses small texture patches around important areas of the face. As such, we have a 3D PDM which contains knowledge about the shape of the face, and 2D texture patches around small sub-regions of the face which replace the holistic texture. The texture patches are built so as to work with as many people as possible in a discriminative approach, and allows us to track the head with less calibration required when changing user or environmental conditions such as lighting. The approach builds the shape model directly in 3D with a full-perspective camera model, whilst building patch models from 2D textures, which firstly constrains the number of possible face combinations in the 2D image (thus improving accuracy) and secondly, by definition, provides the 3D head pose.

3.1 Constructing the 3D shape

Our shape model is based on the well known Point Distribution Model, where the 3D shape is defined as

$$s = (x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n, z_1, z_2, \ldots, z_n) \quad (1)$$

The training data consists, in this case, of 8 annotated 3D meshes taken from a depth camera (where X,Y,Z coordinates are taken by hand from 3D modeling software). Firstly, the 3D data is preprocessed with a Generalised Procrustes alignment where the effects of translation, rotation and scale are minimised. Subsequently, Principal Component Analysis is applied to the shapes to obtain the mean-shape s_0 and a set of n linear basis vectors Φ which describe how the model deforms. This gives the full 3D PDM as

$$s = s_0 + \sum_{i=1}^{n} \Phi_i p_i + \sum_{i=1}^{6} \Psi_i q_i \quad (2)$$

where Φ_i and Ψ_i are the basis vectors for shape deformation and change of pose respectively with p_i and q_i the parameters along those vectors. Since our work aims to reduce computational overhead in real-time calculations on mobile devices, we omit the basis vectors for shape deformation and rely solely on the pose basis vectors, essentially fitting the mean 3d shape to the image. The 6 vectors represent 3 translation parameters (along x, y and z), and three rotation parameters about the x, y and z axes. These 3 parameters can be converted to a traditional rotation matrix via the Rodrigues formula.

For a successful 3D shape tracker, we must choose face points that:

- tend to remain static, so as to reduce the negative effects of a static shape model
- describe the region around the eyes well
- have surrounding textures that are well defined, and are not defined by shadows or outlines.

The final point about outlines in the image is important to emphasise. Many other works attempt to track regions like a persons cheeks from the outline of the face. These points are relatively easy to detect, however they do not represent a single 3D point on the face, but instead a whole region of possible points as the user orients their head about in space. They are thus unsuitable for 3D tracking. We choose 19 specifically selected points to try to fit these criteria. More (or less) points could of course be chosen, but with a trade-off between computational time of the algorithm and accuracy. We have found that 19 points provide decent tracking capabilities while still maintaining high frames per second. We have chosen 3 points on each eyebrow, 4 points around each eye and 5 points around the nose, as shown in Figure 1.

3.2 Creating Texture Patches

To build our discriminative texture patches we use MOSSE filters [Bolme et al. 2010], which act in the Fourier domain. The MOSSE filters are capable of overcoming such issues as minor variances in rotation and large changes in lighting. The texture patches were created from a subset of the Multi-PIE face image dataset [Gross et al. 2010]. We annotated 200 images of 10 individuals (9 males, 1 female; 4 had glasses and all had varying skin-colours and ethinicities) under 20 different lighting conditions. The advantages of these filters include fast training and efficient pointwise multiplication at run-time. To train the filters, small patches were taken from the training images and were affine warped to slightly varying scales, rotations, and translations to create a robust filter. Under experimentation we found that a pixel size of 32x32 was adequate to

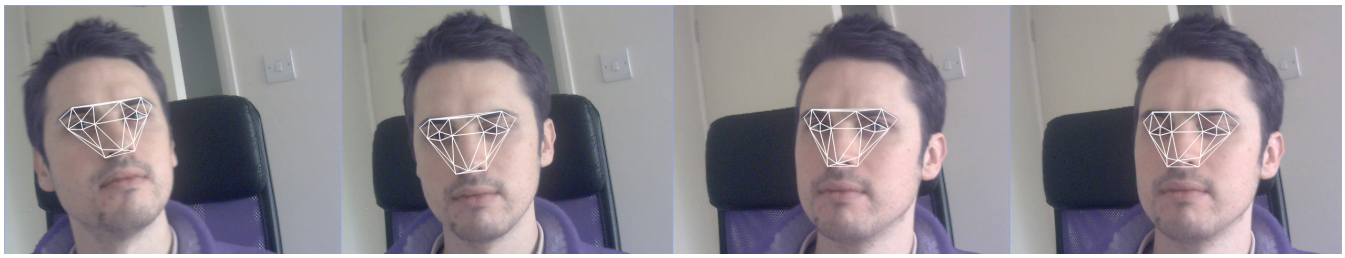

Figure 1: *Examples of the head tracker on a short video sequence. Fast changes in orientation and position temporarily reduce the accuracy of the tracker, particularly at extreme rotations. Loss of texture data for the eye-region during blinks (as seen in the far right image) can also hamper the accuracy of the tracker.*

successfully describe a texture patch. The nature of working in the Fourier domain means that the texture patches need to be in powers of 2. We found 64x64 patches improved detection rates slightly but at the cost of a large increase in computation time. To combat any noise from the MOSSE detectors we perform a post-processing step, where the mean-shift algorithm is used to detect the greatest concentration of high response from the detector. After optimising we denote the best location changes for all patches $i \in 0 \dots n$

$$\Delta Z = \begin{pmatrix} \Delta x_0, & \Delta y_0, & \dots, & \Delta x_n, & \Delta y_n \end{pmatrix} \tag{3}$$

3.3 The Perspective Camera Model

To map our data from 3D to 2D we use a full-perspective camera model. Different webcams have a wide spread of varying intrinsic camera parameters (such as focal length) under which our head tracking model would produce vastly differing and inaccurate estimates, thus making a weak-perspective model unsuitable. We calibrate our camera with a simple checkerboard pattern as standard giving a matrix K which defines the camera intrinsic parameters

$$K = \begin{pmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

where f_x, f_y represent the camera focal length; c_x, c_y represent the camera principal point and α the skew parameter. All 3D measurements in space are defined with the camera at the origin, and are measured in millimetres. We produce homogenous coordinates from a 3D point on the shape by

$$\begin{pmatrix} x_{hom} \\ y_{hom} \\ w_{hom} \end{pmatrix} = K \begin{pmatrix} x_{shape} \\ y_{shape} \\ z_{shape} \end{pmatrix} \tag{5}$$

therefore dividing through by the homogenous value gives the image coordinates:

$$x_{img} = \frac{x_{hom}}{w_{hom}}, \quad y_{img} = \frac{y_{hom}}{w_{hom}} \tag{6}$$

We denote the function for converting a 3D point on the mean shape $X = \{x, y, z\}$ by parameters q to 2D image coordinates as $W(X, q)$.

3.4 Solving for the 3D pose

On the initial frame of a video stream we need to intialise our shape estimate close to the correct values (as we are only tracking a 32x32 square patch for each shape point). We use a simple Haar classfier [Viola and Jones 2001] for face detection to estimate the approximate starting region. For each frame thereafter we use our tracking algorithm only. As we have previously discussed, q represents our 6 pose parameters (3 for translation, 3 for rotation). At any time our 3D shape can be defined as

$$s = s_0 + q\psi \tag{7}$$

where ψ represents our rigid-basis vectors describing movement in translation and rotation and q represents our movement along those transformation from the mean shape. To track the change in 3D pose between frames we need to determine the change in q. However, we first need to know how those parameters relate to the image coordinates, since we only have texture information in 2D. To solve our problem in a least square sense, we need to determine the Jacobian matrix (J) which describes how a small change in each of the parameters q_i is reflected in change of x and y in the image coordinates. Let $X_i = \{x_i, y_i, z_i\}$ be the i^{th} point of the mean shape and q_j be movement along only the j^{th} basis vector

$$J = \begin{pmatrix} \frac{\partial W(X_0,q_0)}{\partial x_{img}} & \frac{\partial W(X_0,q_0)}{\partial y_{img}} & \dots & \frac{\partial W(X_n,q_0)}{\partial x_{img}} & \frac{\partial W(X_n,q_0)}{\partial y_{img}} \\ \frac{\partial W(X_0,q_1)}{\partial x_{img}} & \frac{\partial W(X_0,q_1)}{\partial y_{img}} & \dots & \frac{\partial W(X_n,q_1)}{\partial x_{img}} & \frac{\partial W(X_n,q_1)}{\partial y_{img}} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \frac{\partial W(X_0,q_6)}{\partial x_{img}} & \frac{\partial W(X_0,q_6)}{\partial y_{img}} & \dots & \frac{\partial W(X_n,q_6)}{\partial x_{img}} & \frac{\partial W(X_n,q_6)}{\partial y_{img}} \end{pmatrix} \tag{8}$$

Notice that this formulation has the additional complexity that for each pose in 3D space, this 2D Jacobian is only locally applicable, therefore it has to be recalculated every frame. Additionally, it is only applicable for small rotations. To simplify the creation of this Jacobian matrix in real time we define 6 reference shapes in 3D where

$$\begin{array}{ll} q_{ref1} = (10, 0, 0, 0, 0, 0), & s_{ref1} = s_0 + q_{ref1} * \psi \\ q_{ref2} = (0, 10, 0, 0, 0, 0), & s_{ref2} = s_0 + q_{ref2} * \psi \\ q_{ref3} = (0, 0, 10, 0, 0, 0), & s_{ref3} = s_0 + q_{ref3} * \psi \\ q_{ref4} = (0, 0, 0, 0.1, 0, 0), & s_{ref4} = s_0 + q_{ref4} * \psi \\ q_{ref5} = (0, 0, 0, 0, 0.1, 0), & s_{ref5} = s_0 + q_{ref5} * \psi \\ q_{ref6} = (0, 0, 0, 0, 0, 0.1), & s_{ref6} = s_0 + q_{ref6} * \psi \end{array} \tag{9}$$

The first 3 shapes represent a shift in translation by 10mm in x,y, and z respectively, and the last 3 represent small rotations around the x,y, and z axes. Since the Jacobian values change each frame, it may be required to iterate to a solution (updating the Jacobian matrix several times) if the fps is too low (i.e. if the head moves too much between frames). On each frame we calculate the Jacobian matrix in 2D by projecting these 6 reference shapes and the original meanshape with the current known pose into the image frame. Then we can simply say

$$J_{row_i} = W(s_0, q) - W(s_{ref_i}, q) \tag{10}$$

for each row $i \in 1, ..., 6$ in the Jacobian matrix. With ΔZ being the estimated change in the 2d image (from the MOSSE detectors),

FPS	Eye Corner	Avg pix err x	Avg pix err y	Resets
75	RE / RC	1.73702359	2.37419844	0
	RE / LC	1.63944316	3.11281109	
	LE / RC	3.72725987	2.86814070	
	LE / LC	4.73537683	3.10653377	
60	RE / RC	1.78500342	2.42468500	0
	RE / LC	1.68506277	3.16694117	
	LE / RC	3.68800116	2.90230894	
	LE / LC	4.65725470	3.15000319	
30	RE / RC	1.94616556	2.46617818	1
	RE / LC	1.61177218	3.18407989	
	LE / RC	3.81095982	2.96344233	
	LE / LC	5.00810575	3.32589984	
15	RE / RC	2.83370614	2.43627954	2
	RE / LC	3.02331281	3.21362543	
	LE / RC	5.45969963	3.03827071	
	LE / LC	7.03185415	3.43196487	
10	RE / RC	11.4822721	3.10127211	3
	RE / LC	8.16897297	4.49049807	
	LE / RC	14.7787819	4.48335361	
	LE / LC	23.9565201	4.34037733	

Table 1: *Eye Corner Pixel Errors and Tracking Resets (RE - Right Eye, LE - Left Eye, RC - Right Eye Corner, LC - Left Eye Corner)*

we can then solve for Δq with a least squares algorithm.

$$\Delta q = (JJ^T)^{-1} J \psi \Delta Z \tag{11}$$

4 Experiments

We construct a simple experiment to assess the tracking capabilities of the head tracker. Since 3D data of the head position and orientation is difficult to obtain we analyse our algorithm via another method relevant to this field - tracking of the eye-corners in a video stream. Firstly, we capture a 10-second video stream from a 640x480 webcam (captured at 75 fps, giving 750 image frames) of a user moving their head while sat approximately 60cm away from a monitor screen as shown in Figure 1. The participant was asked to casually move around in their chair while looking at several different regions of the monitor. The camera was fully calibrated beforehand with 75 images of a checkerboard pattern to work out the camera intrinsic values. We hand-annotated the x and y coordinates of all 4 eye corners on all 750 images. We then ran our head tracking algorithm in real-time over the video stream and calculated the average error in x and y pixels. We then reran the experiments at different fps (75,60,30,15,10), skipping frames so that the user's head moves more in between frames and thus increasing the difficulty for the tracking algorithm. On challenging parts of the video, where the head moved and rotated quickly, searching only the local region fails and the tracker is reset with a Haar classifier if required (Resets column - Table 1). Tracking loss was determined by running the Haar Classifier on the estimated face region once per second to check for a positive match.

5 Conclusions

While other approaches attempt to track the head in the 2D image and subsequently attempt to obtain the 3D pose, we have investigated the prospect of building and constraining the shape model directly in 3D with a full-perspective camera model, which firstly constrains the number of possible face combinations in the 2D image (thus improving accuracy) and secondly, by definition, providing the 3D head pose. We provided a simple test case for the common eye tracking problem of successfully locating the eye corners which we achieve on average approximately within 3 pixels when the fps is greater than 15. The eye corners themselves are sometimes even difficult to precisely pick out for a human so this result shows the head tracker is perfectly suited to this task whilst also having further benefits of estimating actual 3D pose from the camera. We see that the biggest issue with our head-tracker is when the user moves their head too quickly (or when fps becomes too low, which causes the same effect). We can counter the tracking loss by reinitialising the tracker on some conditional test. The prototyped 3D model provides a basis of a new approach for a gaze estimation model on mobile devices, by specifically looking at the issues of head-tracking and pose detection. Considerations have been taken to increase usability for different users (through the discriminative patch model approach) and decrease potential errors from lighting (by using MOSSE filters). We now look to make the solution more robust and validate it's accuracy with multiple people under different lighting conditions by using a VICON motion capture system to measure the head position and orientation with high precision. On obtaining a robust head tracker, we will subsequently look at obtaining the gaze point on a screen through looking at the geometric properties of the eye in detail.

References

BOLME, D. S., BEVERIDGE, J. R., DRAPER, B. A., AND LUI, Y. M. 2010. Visual object tracking using adaptive correlation filters. In *CVPR, 2010 IEEE Conference on*, IEEE, 2544–2550.

COOTES, T. F., EDWARDS, G. J., AND TAYLOR, C. J. 2001. Active appearance models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 23*, 6, 681–685.

CRISTINACCE, D., AND COOTES, T. 2006. Feature detection and tracking with constrained local models. In *Proc. British Machine Vision Conference*, vol. 3, 929–938.

GROSS, R., MATTHEWS, I., COHN, J., KANADE, T., AND BAKER, S. 2010. Multi-pie. *Image and Vision Computing 28*.

LANGTON, S. R., HONEYMAN, H., AND TESSLER, E. 2004. The influence of head contour and nose angle on the perception of eye-gaze direction. *Attention, Perception, & Psychophysics 66*.

MARTINS, P., CASEIRO, R., AND BATISTA, J. 2012. Generative face alignment through 2.5 d active appearance models. *Computer Vision and Image Understanding*.

PIRRI, F., PIZZOLI, M., AND RUDI, A. 2011. A general method for the point of regard estimation in 3d space. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE 921–928.*

SESMA, L., VILLANUEVA, A., AND CABEZA, R. *2012. Evaluation of pupil center-eye corner vector for gaze estimation using a web cam. In* Proceedings of the Symposium on Eye Tracking Research and Applications*, ACM, 217–220.*

VALENTI, R., SEBE, N., AND GEVERS, T. *2012. Combining head pose and eye location information for gaze estimation.* Image Processing, IEEE Transactions on*, 99, 1–1.*

VIOLA, P., AND JONES, M. *2001. Rapid object detection using a boosted cascade of simple features. In* CVPR, 2001. IEEE Computer Society Conference on*, vol. 1, IEEE, I–511.*

XIAO, J., BAKER, S., MATTHEWS, I., AND KANADE, T. *2004. Real-time combined 2d+ 3d active appearance models. In* IEEE Computer Vision and Pattern Recognition*, vol. 2.*